1 ***(1 punto)*** *Desde el punto de vista del estado del procesador, explique las diferencias entre el servicio de interrupciones y la concesión de robos de ciclo.*

En los robos de ciclos el procesador se limita a ceder los buses, es decir, a dejar en estado de alta impedancia las líneas de salida que va a usar el módulo de E/S para acceder a memoria. Como no se altera el estado interno del procesador, estas cesiones de buses se pueden hacer casi inmediatamente: tan pronto como el procesador acabe el ciclo de bus en curso o instantáneamente si no lo está usando.

Para servir una interrupción ha de ejecutar una rutina específica y volver al programa interrumpido. Por lo tanto se tienen que servir al finalizar la ejecución de las instrucciones porque en estos casos sólo se necesita el contador de programa para reanudar el programa interrumpido.

Salvaguardar estado.

2 ***(1 punto)*** *Justifique qué técnica de entrada/salida le parece más adecuada para operar una línea serie de un computador con sistema operativo que permite la ejecución concurrente de varios programas (multiprogramación).*

El requisito de multiprogramación descarta la entrada/salida por programa, puesto que en esta técnica se realizan las esperas de forma activa, ejecutando instrucciones, lo impide el paso a ejecución de otros programas. Además, una línea serie es un dispositivo de carácter (no de bloque) y una operación de entrada/salida supone la recepción o transmisión de un único dato. Por lo tanto, la técnica de entrada/salida por DMA tendría que trabajar con bloques de un solo dato y no reduciría el número de interrupciones, aunque exigiría el uso de un módulo de entrada/salida más sofisticado. En conclusión, la técnica más adecuada es la entrada/salida mediante interrupciones.

3 ***(1 punto)*** *Enumere las diferencias entre las secuencias de reconocimiento de interrupción para la identificación del peticionario mediante muestreo (polling) y vectorización.*

La principal diferencia es que mediante vectorización se utiliza un ciclo de bus de reconocimiento de interrupciones, que tiene por objetivo que el módulo de entrada/salida solicitante se identifique. Este ciclo de bus está caracterizado por la activación de la línea de reconocimiento de interrupciones (INTA) y es un ciclo especial de lectura para obtener el identificador (vector) del módulo solicitante de la interrupción, o del más prioritario en caso de múltiples solicitudes.

4 *Se desea construir un servidor de backups con un procesador de 32 bits y capacidad de procesamiento de 1.000 MIPS cuya secuencia de reconocimiento de interrupciones dura 8ñs.*

Los discos duros que se van a usar tienen las siguientes características:

- *Velocidad de transferencia:* $40 \cdot 10^6$ *bytes/s.*
- *Tiempo medio de acceso: 5 ms.*
- *Tamaño del sector: 512 bytes.*

Los módulos de E/S de los discos duros operan mediante interrupciones con los siguientes parámetros:

- *Las rutinas de inicio y fin de una operación de E/S constan de 50 y de 75 instrucciones respectivamente.*
- *Su rutina de tratamiento de las interrupciones consta de 40 instrucciones.*
- *Buffer de 2 registros de datos de 32 bits.*

a) (1 punto) *Calcule cuánto* **tiempo** *de CPU se ocupa en una operación de estos discos duros.*

b) (1 punto) *Calcule el número máximo de discos duros que pueden operar simultáneamente con este procesador.*

Como el número de discos duros que pueden operar simultáneamente no se considera suficiente, se decide que los módulos de E/S operen por DMA con las siguientes características:

- *El protocolo de concesión y liberación de los buses dura 4 ns en total.*
- *El ciclo de acceso a memoria tiene una duración de 2 ns.*
- *Las rutinas de inicio y fin de una operación de E/S constan ahora de 100 y de 150 instrucciones respectivamente.*

c) (1 punto) *Calcule el* **porcentaje de tiempo** *de CPU que se ocupa en una operación de estos discos duros.*

d) (1 punto) *Calcule el número máximo de discos duros que pueden operar simultáneamente con este procesador.*

Un cliente del servidor de backups solicita un archivo que ocupa 5 kbytes en sectores no consecutivos de uno de los discos duros. El archivo se transmite por una red Ethernet que opera por DMA con los siguientes parámetros:

- *Velocidad de transmisión* $1 \cdot 10^9$ *bits/s.*
- *Las rutinas de inicio y fin de una operación de E/S constan de 50 y de 75 instrucciones respectivamente.*
- *Registro de datos de 32 bits.*
- *Tamaño de bloque de 1 kbytes.*

e) (2 puntos) *Si ambos dispositivos operan simultáneamente, calcule en qué instante terminan las operaciones para la lectura y transmisión del fichero.*

f) (1 punto) *Calcule qué porcentaje de tiempo queda libre para la CPU durante dichas operaciones.*

SOLUCIÓN

a) La CPU está ocupada en operaciones relacionadas con esta operación de E/S durante las rutinas de inicio y finalización y durante el tiempo dedicado a reconocer y tratar las interrupciones, en total:

$$\begin{aligned} t_{cpu} &= t_{ini} + (N_{int} - 1) \times t_{int} + t_{última-int} = t_{ini} + N_{int} \times (t_{sri} + t_{rti}) + t_{fin} = \\ &= 50\ ns + \frac{512\ bytes}{2 \cdot 4\ bytes} \times (8\ ns + 40\ ns) + 75\ ns = 50\ ns + 3.072\ ns + 75\ ns = 3.197\ ns \end{aligned}$$

b) Se calcula la capacidad de procesamiento necesaria para atender las interrupciones de un disco.

$$CP_{HD} = Freq_{int} \times I_{int} = \frac{40 \cdot 10^6\ bytes/s}{2 \cdot 4\ bytes} \times (8\ instr + 40\ instr) = 5 \cdot 10^6 \times 48\ instr/s = 240\ MIPS$$

Luego pueden operar simultáneamente 1.000 MIPS/240 MIPS = 4 discos duros.

c) Se calcula el tiempo de CPU ocupada y el tiempo total de una operación del disco.

$$\begin{aligned} t_{op} &= t_{ini} + t_{acc} + t_{trans} + t_{último-robo-ciclo} + t_{int} = t_{ini} + t_{acc} + t_{trans} + t_{último-robo-ciclo} + (t_{sri} + t_{fin}) = \\ &= 100\ ns + 5 \cdot 10^6\ ns + \frac{512\ bytes}{40 \cdot 10^6\ bytes/s} + (4\ ns + 2 \cdot 2\ ns) + (8\ ns + 150\ ns) = \\ &= 100\ ns + 5.000.000\ ns + 12.800\ ns + 8\ ns + 158\ ns = 5.013.066\ ns \\ t_{cpu} &= t_{ini} + N_{robo-ciclo} \times t_{robo-ciclo} + t_{int} = \\ &= 100\ ns + \frac{512\ bytes}{2 \cdot 4\ bytes} \times 8\ ns + (8\ ns + 150\ ns) = 770\ ns \end{aligned}$$

En consecuencia, el porcentaje de ocupación de la CPU por la operación del disco es:

$$\%_{ocupada} = \frac{t_{ocupada}}{t_{total}} \times 100 = \frac{770\ ns}{5.013.066\ ns} \times 100 = 0.015\ \%$$

d) Se calcula la capacidad de procesamiento necesaria para atender los robos de ciclo de un disco. Puesto que cada robo de ciclo requiere 4ns para ceder y recuperar los buses y otros 4ns para transferir las dos palabras del buffer, se emplean 8ns por cada robo de bus, es decir, el tiempo equivalente a la ejecución de 8 instrucciones:

$$CP_{HD} = Freq_{robos} \times I_{robo} = \frac{40 \cdot 10^6\ bytes/s}{2 \cdot 4\ bytes} \times 8\ instr = 5 \cdot 10^6 \times 8\ instr/s = 40\ MIPS$$

Luego pueden operar simultáneamente 1.000 MIPS/40 MIPS = 25 discos duros.

e) Se calcula el tiempo de una operación de la red Ethernet.

$$\begin{aligned} t_{op-Ethernet} &= t_{ini} + t_{trans} + t_{último-robo-ciclo} + t_{última-int} = \\ &= 50\ ns + \frac{1.024\ bytes \times 8\ bits/bytes}{10^9\ bits/s} + (4\ ns + 2\ ns) + (8\ ns + 75\ ns) = \\ &= 50\ ns + 8.192\ ns + 6\ ns + 83\ ns = 8.331\ ns \end{aligned}$$

El tiempo total invertido en la lectura y transmisión del fichero será el correspondiente a $5 \times 1024\ bytes/512\ bytes =$ 10 operaciones del disco duro más la última operación de la red Ethernet.

$$t_{total} = 10 \times t_{op-HD} + t_{op-Ethernet} = 10 \times 5.013.066\ ns + 8.331\ ns = 50.138.991\ ns$$

f) Se calcula el tiempo de CPU de una operación (1 kb) de la red Ethernet.

$$\begin{aligned} t_{cpu} &= t_{ini} + N_{robo-ciclo} \times t_{robo-ciclo} + t_{int} = \\ &= 50\ ns + \frac{1.024\ bytes}{4\ bytes} \times 6\ ns + (8\ ns + 75\ ns) = 1.669\ ns \end{aligned}$$

El tiempo de CPU libre será el tiempo total menos el correspondiente a 10 operaciones del disco y 5 de la red Ethernet.

$$\begin{aligned} t_{cpu\_ocupada} &= 10 \times t_{cpu-HD} + 5 \times t_{cpu-Ethernet} = 10 \times 770\ ns + 5 \times 1.669\ ns = 16.045\ ns \\ t_{cpu\_libre} &= t_{total} - t_{cpu\_ocupada} = 50.138.991\ ns - 16.045\ ns = 50.122.946\ ns \end{aligned}$$

$$\%_{cpu-libre} = \frac{t_{ocupada}}{t_{total}} \times 100 = \frac{50.122.946\ ns}{50.138.991\ ns} \times 100 = 99.96\ \%$$

1 *(1 punto) Describa la diferencia entre la tasa de éxito local y global para memorias cache multinivel.*

SOLUCIÓN

- La tasa de éxito local para un nivel dado se define como el cociente entre el número de aciertos que se produce en dicho nivel y el número de accesos realizados a ese mismo nivel.
- La tasa de éxito global se define, para un nivel dado, como el cociente entre el número de aciertos que se produce en cualquier nivel de memoria, desde el más cercano al procesador al que se está considerando, y el número total de accesos emitidos por el procesador.

En el caso de la cache de primer nivel ambas tasas coinciden, puesto que todos los accesos emitidos por el procesador pasan por ella.

Para evaluar la eficiencia de la memoria cache multinivel se suele utilizar la tasa de éxito global, puesto que indica la eficacia de la memoria cache indicando realmente el porcentaje de los accesos totales que no van a memoria principal.

2 *(6 puntos) Se tiene un computador de 32 bits que dispone de caches separadas para instrucciones y datos con las siguientes características:*

- *Capacidad de cada memoria cache: 16 Kbytes*
- *Tamaño de los bloques de cache: 16 bytes*
- *Organización asociativa por conjuntos de 2 bloques*
- *Política de reemplazo FIFO* (First Input First Output)
- *Política de lectura: OOF* (Out of Order Fetch)
- *Política de escritura de la cache de datos: diferida con actualización* (CBWA: Copy Back With Allocation)
- *En los fallos en escritura se modifica primero la palabra en memoria principal y posteriormente se actualiza el bloque en la cache.*
- *Tiempo de acceso: 2ñs.*

Este computador dispone de una memoria principal de 8 módulos con entrelazado simple de orden inferior. El tiempo de acceso a memoria principal para leer o escribir una palabra es 40ñs y el tiempo necesario para leer o escribir un bloque de 4 palabras 60ñs.

Se está ejecutando en este computador el siguiente fragmento de un programa:

```
for (i=0; i<2048; i++)      /* 2.048 iteraciones */
   c[i] = a[i] + b[i];
```

Figura 1 Fragmento de programa original.

Cada elemento de los vectores `a`, `b` *y* `c` *ocupa una palabra y están ubicados en las direcciones* `Df(a) = 00002000H`, `Df(b) = 00004000H` *y* `Df(c) = 00006000H`. *Las memorias cache están inicialmente invalidadas.*

a) *(1 punto) Calcule en qué conjuntos de la cache de datos se ubicarán los bloques de los vectores* `a`, `b` *y* `c` *y la tasa de aciertos de la cache de datos para la ejecución de este fragmento de programa.*

Como la tasa de aciertos es inusitadamente baja se estudian dos alternativas para mejorar esta tasa de aciertos. La primera consiste en reescribir el fragmento del programa:

```
for (i=0; i<2048; i++)      /* 2.048 iteraciones */
   c[i] = a[i];
for (i=0; i<2048; i++)      /* 2.048 iteraciones */
   c[i] = c[i] + b[i];
```

Figura 2 Fragmento de programa modificado.

b) (1 punto) Calcule la tasa de aciertos de la cache de datos para la ejecución de este nuevo fragmento de programa (figura 2) y el número de bloques modificados que son desalojados.

c) (1 punto) Calcule el tiempo total de acceso a memoria para datos en la ejecución de este nuevo fragmento de programa (figura 2).

d) (1 punto) Calcule el tiempo total de ocupación del sistema de memoria debido a los accesos a datos en la ejecución de este nuevo fragmento de programa (figura 2).

La segunda alternativa consiste en reubicar el vector `c` *en la dirección* `Df(c) = 00007000H`.

e) (1 punto) Considere la ejecución del fragmento de programa original (figura 1) con la nueva ubicación del vector `c`. *Calcule para este caso la tasa de aciertos de la cache de datos y el número de bloques modificados que son desalojados.*

(1 punto) Calcule el tiempo total de acceso a memoria debido a los accesos a datos en la ejecución del fragmento de programa original (figura 1) con la nueva ubicación del vector `c`.

SOLUCIÓN

a) Para resolver este apartado hay que calcular cómo se interpretan las direcciones físicas por la memoria cache. Como es asociativa por conjuntos habrá que calcular el número de conjuntos.

$$Núm\_conjuntos = \frac{16\ KB}{2\ Bq/Conjunto \cdot 16\ bytes/Bq} = 512\ Conjuntos$$

Por lo tanto las direcciones físicas de los primeros bloques de `a`, `b` y `c` se interpretan:

31 13	12 4	3 0	
Etiqueta	Conjunto	Byte	
0...001	0...0	0000	`Df(a)`
0...010	0...0	0000	`Df(b)`
0...011	0...0	0000	`Df(c)`

Para los últimos bloques hay que calcular cuántos bloques ocupan los vectores:

$$Núm\_bloques = \frac{2.048 \cdot 4\ bytes}{16\ bytes/Bq} = 512\ bloques$$

El número de bloques coincide con el número de conjuntos de la memoria cache. Por lo tanto, si los primeros bloques de los vectores van al conjunto 0, los últimos irán al conjunto 511.

Es decir en cada iteración del bucle se hace referencia a 3 bloques que se ubican en el mismo conjunto. Como los conjuntos son de 2 bloques ocurrirá lo siguiente en las dos primeras iteraciones del bucle:

Iteración	i=0			i=1		
Referencia	a[0]	b[0]	c[0]	a[1]	b[1]	c[1]
Caché	Fallo	Fallo	Fallo	Fallo	Fallo	Fallo
Reemplazo	No	No	a[0..3]	b[0..3]	c[0..3]	a[0..3]

Al repetirse este patrón de fallos y desalojos resulta una tasa de aciertos nula.

b) En este caso durante la ejecución del primer bucle únicamente se producen fallos forzosos, ya que se evitan los conflictos. Así tendremos 512 fallos forzosos para el vector `a` y otros 512 fallos forzosos para el vector `c`. La cache de datos se llena con los 512 bloques de vector `a` y los 512 bloques de vector `c`.

La siguiente tabla muestra lo que ocurre durante las primeras iteraciones del segundo bucle:

Iteración	i=0			i=1		
Referencia	c[0]	b[0]	c[0]	c[1]	b[1]	c[1]
Caché	Acierto	Fallo	Acierto	Acierto	Acierto	Acierto
Reemplazo	-	a[0..3]	-	-	-	-

Es decir, durante la ejecución del segundo bucle sólo se producen 512 fallos forzosos para el vector `b`. Por lo tanto la tasa de acierto es:

$$M_r = \frac{Núm\_fallos}{Núm\_accessos} = \frac{2 \cdot 512 + 512}{2.048 \cdot 2 + 2.048 \cdot 3} = 0,15 \rightarrow H_r = 0,85$$

Nótese que se desalojan los bloques del vector `a` ya que son los más antiguos y que estos bloques no están modificados.

c) El tiempo de acceso a memoria será la suma del tiempo de los aciertos más el tiempo de los fallos. En estos fallos no se sustituyen bloques modificados y como la lectura es OOF se considera el tiempo de acceso a memoria principal de una sola palabra.

$$t_{aciertos} = Núm\_aciertos \times t_{cache} = (2.048 \cdot 5 - 3 \cdot 512) \times 2\ ns = 17.408\ ns$$

$$t_{fallos} = Núm\_fallos \times (t_{cache} + t_{Mpal}) = 3 \cdot 512 \times (2\ ns + 40\ ns) = 64.512\ ns$$

$$t_{total} = 17.408\ ns + 64.512\ ns = 81.920\ ns$$

d) Para calcular el tiempo de ocupación del sistema de memoria hay que considerar el tiempo de lectura del bloque completo en vez del de una palabra en los accesos de lectura con fallo. En los fallos en escritura se modifica primero la palabra en memoria principal y posteriormente se actualiza el bloque en la cache por lo tanto habrá que añadir el tiempo de lectura del bloque completo al tiempo de acceso.

$$t_{fallos\_lectura} = Núm\_fallos \times (t_{cache} + t_{Mbq}) = 2 \cdot 512 \times (2\ ns + 60\ ns) = 63.488\ ns$$

$$t_{fallos\_escritura} = Núm\_fallos \times (t_{cache} + t_{Mpal} + t_{Mbq}) = 512 \times (2\ ns + 40\ ns + 60\ ns) = 52.224\ ns$$

$$t_{total} = 17.408\ ns + 63.488\ ns + 52.224\ ns = 133.120\ ns$$

e) Habrá que calcular en qué conjuntos de la cache se ubica ahora el vector `c`.

31 13	12 4	3 0	
Etiqueta	Conjunto	Byte	
0...011	10...0	0000	`Df(c)`

El vector `c` se ubica a partir del conjunto 256 y el último bloque del vector se ubica en el conjunto 255. Nótese que el bloque 255 del vector `c` se ubica en el último conjunto (511) y el bloque 256 del vector `c` en el primer conjunto (0).

Es decir en cada iteración del bucle se hace referencia a 3 bloques que ahora no se ubican en el mismo conjunto y se evitan los conflictos. En las primeras iteraciones del bucle ocurrirá lo siguiente:

Iteración	i=0			i=1		
Referencia	a[0]	b[0]	c[0]	a[1]	b[1]	c[1]
Caché	Fallo	Fallo	Fallo	Acierto	Acierto	Acierto
Reemplazo	No	No	No	No	No	No

Tras las primeras 1.024 iteraciones, la primera mitad del vector `c` se ha ubicado en los conjuntos [256..511] y las primeras mitades de los vectores `a` y `b` en los conjuntos [0..255]. Por lo tanto, en las siguientes iteraciones del bucle ocurre lo siguiente:

Iteración	i=1.024			i=1.025		
Referencia	a[1.024]	b[1.024]	c[1.024]	a[1.025]	b[1.025]	c[1.025]
Caché	Fallo	Fallo	Fallo	Acierto	Acierto	Acierto
Reemplazo	No	c[0..3]	a[0..3]	No	No	No

Es decir, únicamente se producen fallos forzosos y se desalojan 512 bloques de los cuales 256 bloques son del vector `a` y otros 256 bloques son del vector `c` que están modificados. Por lo tanto la tasa de acierto es:

$$M_r = \frac{Núm\_fallos}{Núm\_accessos} = \frac{3 \cdot 512}{2.048 \cdot 3} = 0,25 \rightarrow H_r = 0,75$$

f) El tiempo de acceso a memoria será la suma del tiempo de los aciertos más el tiempo de los fallos. Además 256 fallos sustituyen bloques del vector `c` que están modificados.

$$t_{aciertos} = Núm\_aciertos \times t_{cache} = (2.048 \cdot 3 - 3 \cdot 512) \times 2\ ns = 9.216\ ns$$

$$t_{fallos} = Núm\_fallos \times (t_{cache} + t_{Mpal}) + Núm\_reemplazos \times t_{Mbq} = 3 \cdot 512 \times (2\ ns + 40\ ns) + 256 \times 60\ ns = 79.872\ ns$$

$$t_{total} = 9.216\ ns + 79.872\ ns = 89.088\ ns$$

3 *(3 puntos) Considere ahora que al mismo computador del ejercicio 2 se le dota de memoria virtual con las siguientes características:*

- *Espacio virtual direccionable: 64 TB.*
- *Tres niveles de tablas de páginas.*
- *Las entradas de las tablas de páginas ocupan una palabra.*
- *Sendas TLBs para datos y instrucciones, con un tiempo de acceso de 1 ns.*

a) (1 punto) Indique cómo se interpretan las direcciones virtuales sabiendo que se puede solapar la traducción de direcciones en las TLBs con el acceso a las memorias cache y que cada tabla de páginas ocupa una página.

b) (1 punto) Calcule los tiempos mínimo y máximo de acceso al sistema de memoria, suponiendo de nuevo que no se produce ningún fallo de página.

c) (1 punto) Indique cuántas páginas se necesitan para ubicar y traducir las direcciones correspondientes al vector `c` si se coloca en la dirección virtual `Dv(c) = 3FFFFF007000H`.

SOLUCIÓN

a) Para que se pueda solapar los accesos a las TLBs y memorias cache, basta con que el campo desplazamiento, que no se traduce, sea igual o mayor que los campos conjunto y byte. Es decir necesitaríamos unas páginas de un tamaño mínimo de 2^{13} *bytes* $= 8\ KB$.

Si las páginas son de 8 KB y las entradas de las tablas de páginas son de una palabra (4 B), las tablas de página de primer nivel tendrán 2.048 entradas. De este modo las direcciones virtuales se interpretarían así:

45 — 35	34 — 24	23 — 13	12 — 0
Entrada PVN1	Entrada PVN2	Entrada PVN3	Desplazamiento
11 bits	11 bits	11 bits	13 bits

b) Para calcular el tiempo máximo de acceso, se calculará el tiempo máximo de traducción y el tiempo máximo de acceso a la información.

El tiempo máximo de traducción se produce cuando hay un fallo en la TLB y hay que traducir en los 3 niveles de página de memoria principal.

$$t_{máximo\_traducción} = 1\ ns + 3 \times 40\ ns = 121\ ns$$

El tiempo máximo de acceso a la información se produce al acceder a datos, cuando hay fallo en la memoria cache y el bloque a desalojar está modificado:

$$t_{máximo\_información} = 2\ ns + 60\ ns + 40\ ns = 102\ ns$$

Así se obtiene:

$$t_{máximo\_acceso} = 121\ ns + 102\ ns = 223\ ns$$

El tiempo mínimo es el tiempo de acceso a la memoria cache (2ñs) puesto que podemos solapar la traducción en TLB con el acceso a la memoria cache.

c) El vector `c` tiene 2.048 elementos de 4 bytes por lo tanto tiene un tamaño de 8 KB, es decir de una página. Entonces para ubicar el vector `c` hace falta una página si su dirección base es múltiplo del tamaño de página. En caso contrario serán necesarias dos páginas.

`Dv(c)` se interpreta por la MMU de la siguiente forma:

45 35	34 24	23 13	12 0
Entrada PVN1	Entrada PVN2	Entrada PVN3	Desplazamiento
1...1	1...1	0...011	10...0
11 bits	11 bits	11 bits	13 bits

Como `Dv(c)` no está "alineada" a página, se necesitan dos páginas para ubicar el vector `c`. Además, solo se necesitan tres páginas para traducir sus direcciones (para almacenar una tabla de primer nivel, una de segundo y una de tercer nivel). Nótese que en la situación más desfavorable un único par de páginas consecutivas comenzando en una `Dv(c)` distinta podría necesitar cuatro o hasta cinco páginas para traducir las direcciones del vector `c`: una para la tabla de primer nivel, dos páginas para almacenar sendas tablas de segundo nivel y otras dos para almacenar las dos tablas de tercer nivel que en tal caso se requerirían.